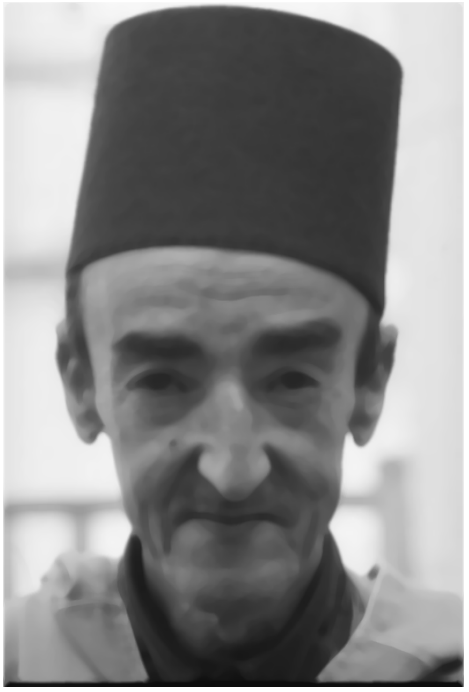
Computer Vision
Assignment 1

Name:	ID:
Manar Amr Mohammed	18011842
Nour Waled Ali	18012006
Youssef Mohamed Hussein	18015031

Part1: Applying Image Processing Filters For Image Cartoonifying

- *Color to grey:*


- *Smoothing the image:* using a median smoothing filter k=11

- *Edge detection (Laplacian filter)*

- *Inverse binary thresholding (at 4,255)*

- *Bilateral filter(d=20,sigmaColor=40,sigmaSpace=70)*

- *Input vs Output*

Part2: Road Lane Detection Using Hough Transform

- *Smoothing the image:* using a median smoothing filter

Image1: The kernel size is 5 (positive odd integer)

original

blurred

Image2: The kernel size is 3 (positive odd integer)

original

blurred

- *Edge Detection:*

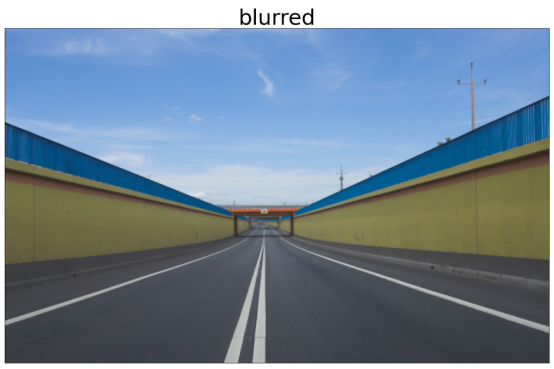
  Edge Detection is performed using Canny's algorithm on the smoothed images.

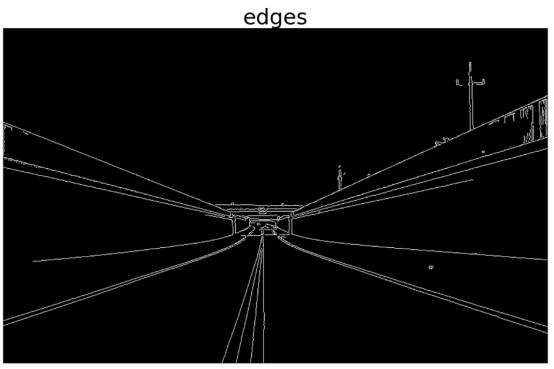
  Double Thresholding:

  - Canny's algorithm compares gradient values against 2 thresholds, lower and upper.
  - The gradients that are smaller than the low threshold value are suppressed.
  - The gradients higher than the high threshold value are marked as strong ones and the corresponding pixels are included in the final edge map.
  - The gradients that lie between lower and upper thresholds are marked as weak edges. Weak edges are either suppressed or included in the final edge map depending on the neighboring edges.

Image1: The threshold values are (100, 200)

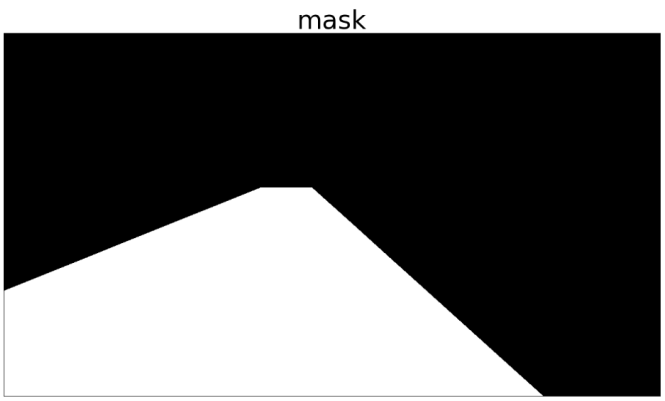
Image2: The threshold values are (100, 200)





- *Region Of Interest:* focuses on necessary edges (on the road), defines a polygon with the size of interesting points to mask the noise edges, then produces only a necessary edge image.

Image1



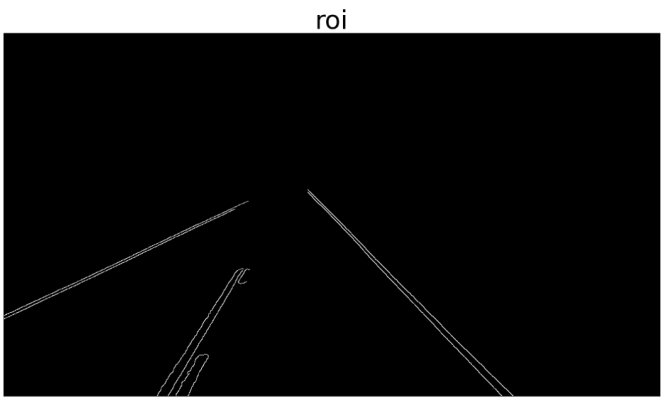
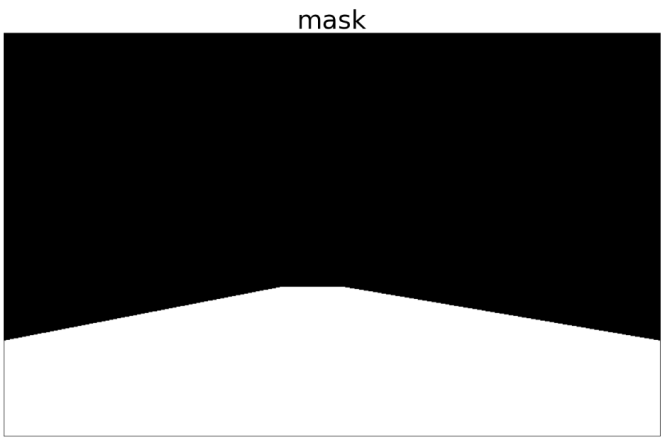
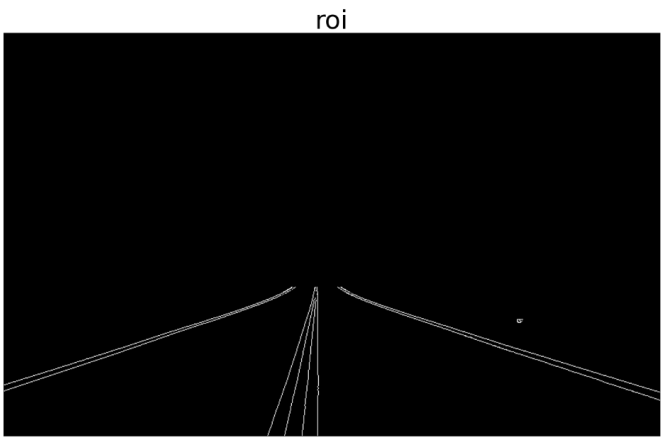


Image2





- *Accumulation into (ρ,θ) space using Hough transform:*
  - The space of ρ and θ is discretized where ρ ∈ [-img_diag, img_diag] and θ ∈ [0°, 180°].
  - each parameter pair (ρ, θ) in the hough space represents a line in the image space.
  - An array of zeros (Accumulation array) is initialized with the shape (ρ, θ) to accumulate the votes of edge points for each line.

```
init accumulator[2*diag_len, 180] with zeros
for each non-zero pixel in the detected edges image:
    x, y = pixel coordinates
    for t in range(0, 180):
        rho = round(x * cos(t) + y * sin(t)) + diag_len
        accumulator[rho, t] += 1
```

  - The value of ρ is shifted by the length of the image diagonal to fit the array indexing.
  - The Hough Space plots show the main points of intersection that represent the lines with the highest votes.

Image1: Hough Space shows 3 main points of intersection.

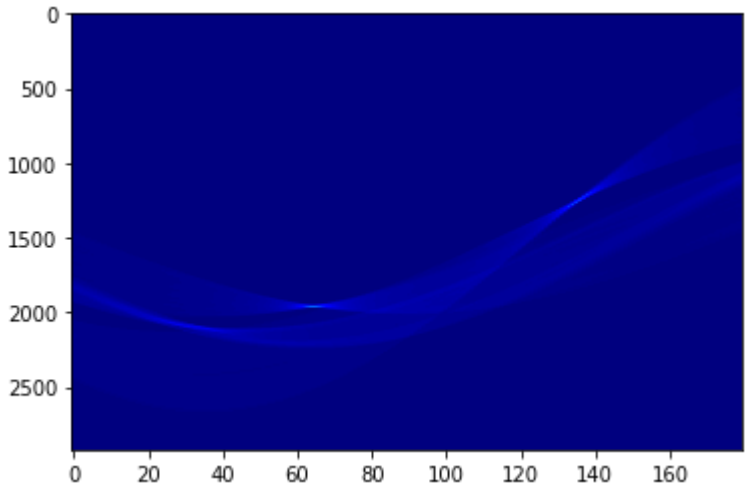
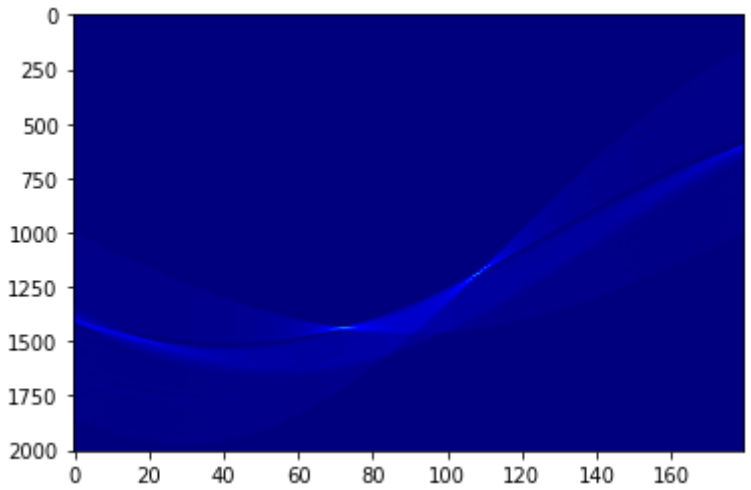


Image2: Hough Space shows 4 main points of intersection.



- *Refining Coordinates:*
    1. Using Thresholding:
        - all lines with votes less than the specified threshold are suppressed. Their votes in the accumulation array are set to zeros.

Image1: All values less than 200 are set to 0.

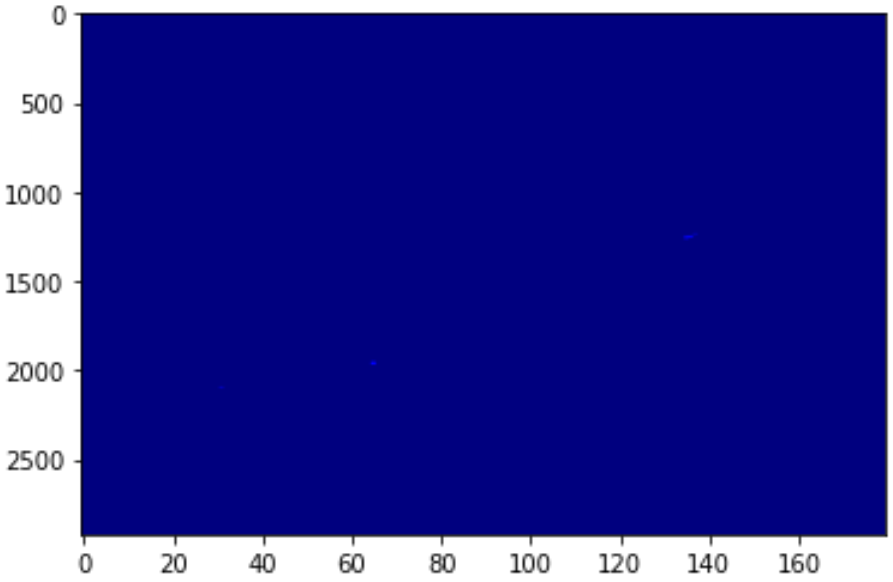
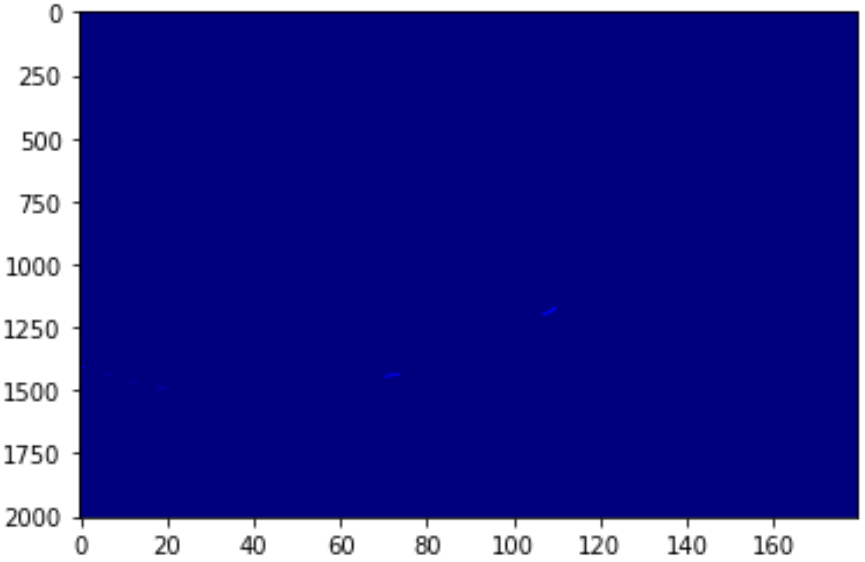


Image2: All values less than 100 are set to 0.

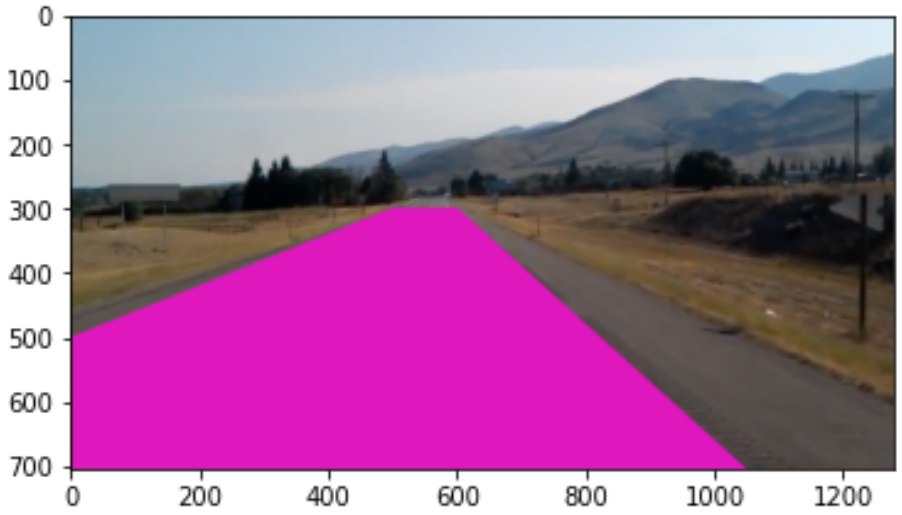


2. Non Max Suppression:
    - filters out the lines with the highest votes within a window (local maxima) and suppresses all other lines.
    - It selects the maximum line out of many overlapping lines in the image.

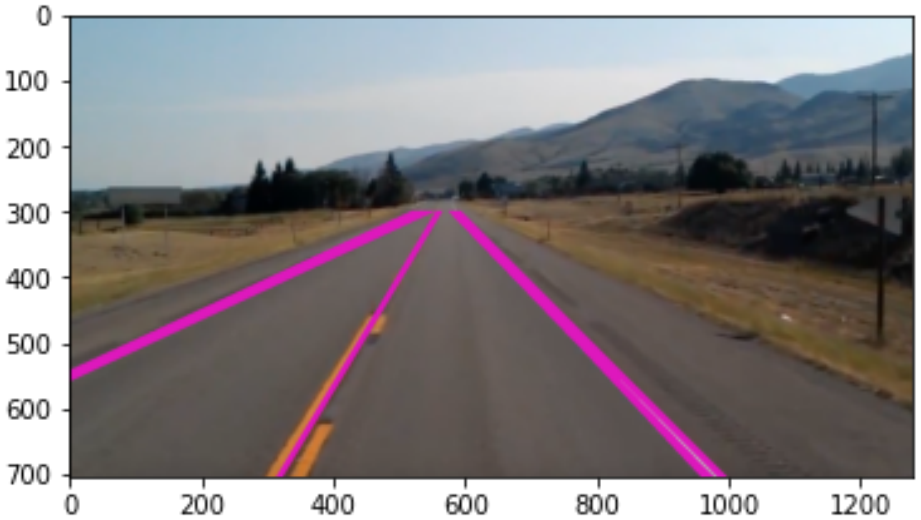
- *Results:*

Image1:

before thresholding



after thresholding & before non-max suppression



after non-max suppression (final result)

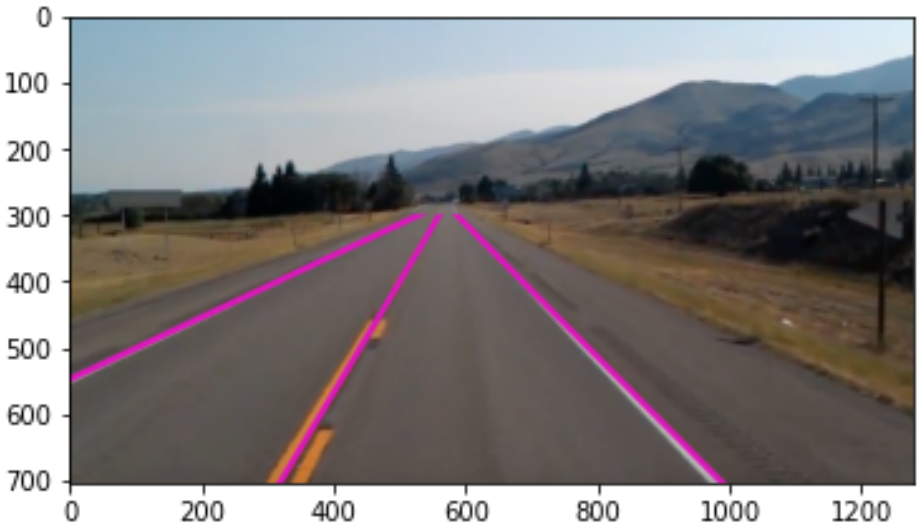
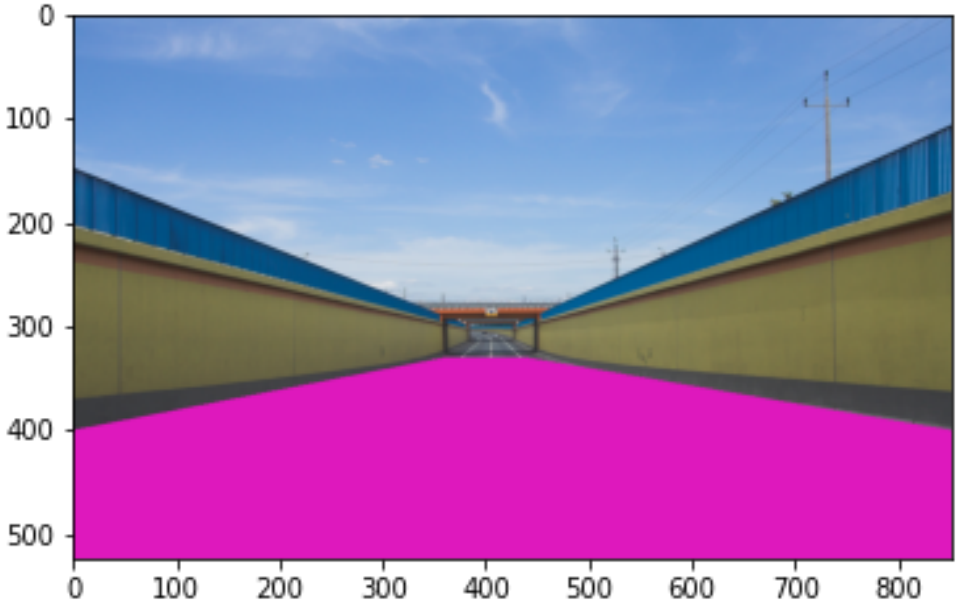
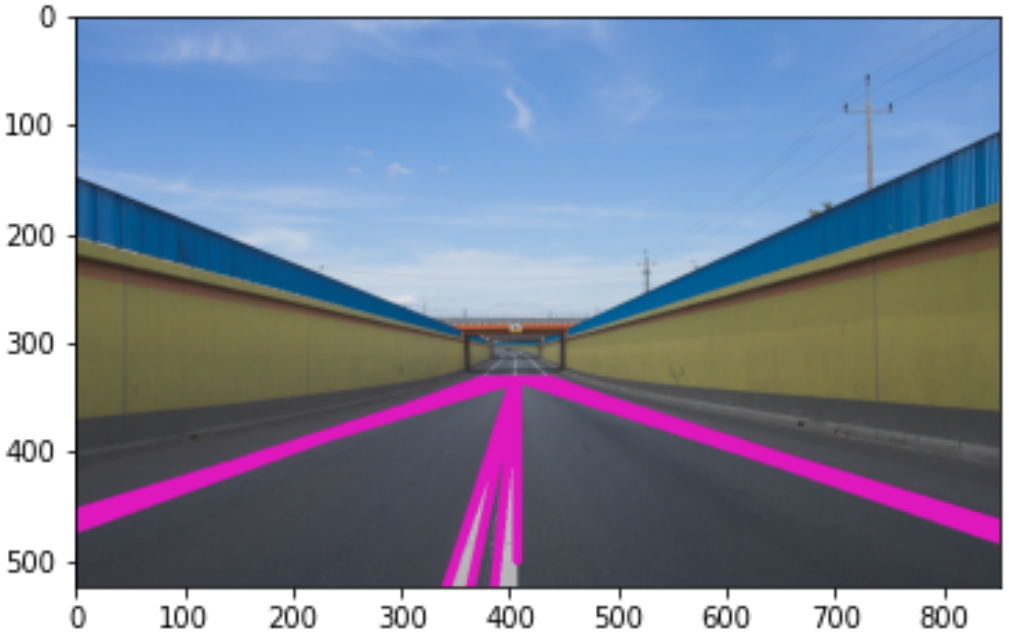


Image2:

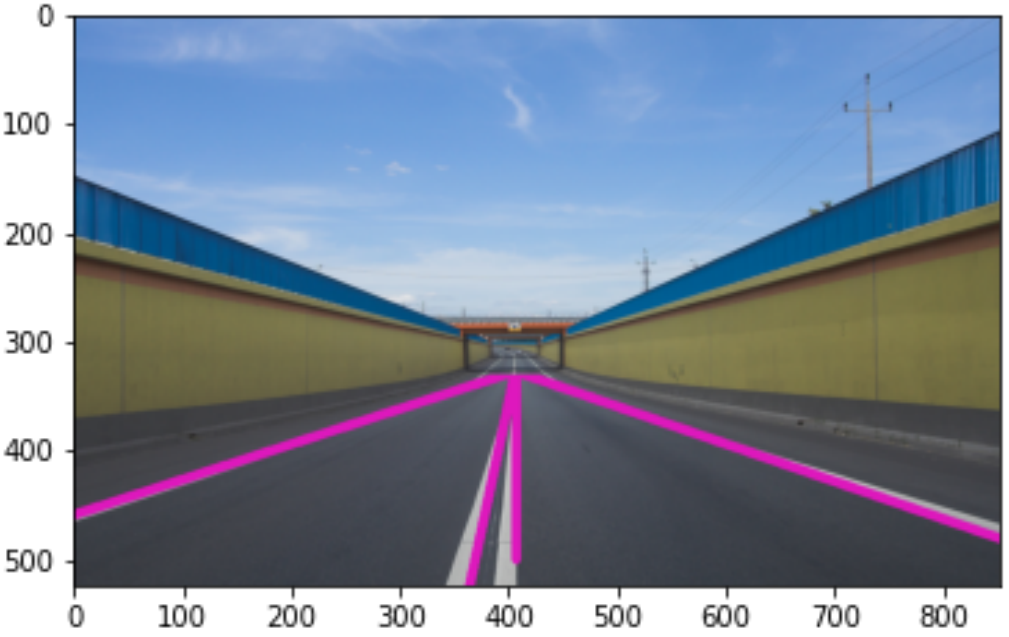
before thresholding



after thresholding & before non-max suppression



after non-max suppression (final result)



- *References:*
  Canny Algorithm
  Hough Transform
  Non Max Suppression
  Drawing lines on images with cv2
  Images overlaying
- *Colab Notebook: link*